Installare LARAVEL

*composer global require laravel/installer*

Nella cartella del progetto

*Laravel new fumetti* (fumetti è il nome del progetto)

Impostare il virtual server su questa cartella creata da laravel:

*fumetti/public*

Modello MVC (model viewer controller)

Il controller contiene la logica alla creazione della pagina

Il viewer mostra la pagina

Ad ogni url è associata una riga di routing che chiama una funzione del controller (che a sua volta prende i dati usando i model che a sua volta chiama la parte di view.

Nella cartella routes c’è il file web.php.

Inserire nel file la seguente riga:

*Route::get('fumetti/create', 'FumettiController@create')->name('fumetti.create');*

Creare un controller

*php artisan make:controller FumettiController*

abbiamo creato il controller che abbiamo definito nel file web.php. Il controller viene creato nella cartella app/http/controllers (file FumettiController.php)

Il controller è una classe.

Inseriamo all’interno della classe il seguente codice:

*public function create(){*

*return view('fumetti.create');*

*}*

Creare una view

Nella cartella resources/views creiamo una cartella con il nome del progetto (fumetti)

Nella cartella fumetti creiamo il file create.blade.php (qui ci andrebbe il codice della pagina)

Ora chiamando la pagina locale.fumetti.it/create

Si vede il contenuto del file create.blade.php

Ricapitolando:

Abbiamo definito una routes nel file routes/web.php

Che prende un url (fumetti/create) che verrà gestita dall’applicazione

La funzione name associa un nome all’url che permetterà di chiamarla

E associa anche un controller che gestirà la richiesta

Il framework prende la url e chiama la funzione

Per ora nella funzione richiamiamo semplicemente una view

Query:

*$autori = DB::table('rel\_titolo\_autore\_ruolo')->where('titolo\_id', '=', $id\_titolo)*

*->join('autore', 'autore.id', '=', 'rel\_titolo\_autore\_ruolo.autore\_id')*

*->join('ruolo', 'ruolo.id', '=', 'rel\_titolo\_autore\_ruolo.ruolo\_id')*

*->get(['autore.id AS autore\_id', 'autore.nome', 'autore.cognome', 'ruolo.id AS ruolo\_id', 'ruolo.descrizione']);*

La variabile $autori è un vettore dove in ogni posizione c’è un vettore che contiene i seguenti dati:

* Autore\_id
* Autore.nome
* Autore.cognome
* Ruolo\_id
* Ruolo.descrizione

Creazione dell’array $info\_autori

Questo vettore conterrà una serie di autori e per ogni autore è associato un vettore di ruoli

foreach($autori AS $current\_autore){

if(!isset($info\_autori[$current\_autore->autore\_id])){

$info\_autori[$current\_autore->autore\_id] = [];

$info\_autori[$current\_autore->autore\_id]['nome'] = $current\_autore->nome . ' ' . $current\_autore->cognome;

$info\_autori[$current\_autore->autore\_id]['ruoli'] = [];

}

$info\_autori[$current\_autore->autore\_id]['ruoli'][$current\_autore->ruolo\_id] = $current\_autore->descrizione;

}

foreach($autori AS $current\_autore){

per ogni posizione del vettore $autori restituito dalla query

if(!isset($info\_autori[$current\_autore->autore\_id])){

se è la prima volta che inserisco l’autore ($current\_autore->autore\_id)

$info\_autori[$current\_autore->autore\_id] = [];

$info\_autori[$current\_autore->autore\_id]['nome'] = $current\_autore->nome . ' ' . $current\_autore->cognome;

$info\_autori[$current\_autore->autore\_id]['ruoli'] = [];

dico che $info\_autori[$current\_autore->autore\_id] è un vettore fatto così:

**$info\_autori[$current\_autore->autore\_id] = [**

**nome => ‘nome e cognome dell’autore corrente’**

**ruoli => []**

**]**

$info\_autori[$current\_autore->autore\_id]['ruoli'][$current\_autore->ruolo\_id] = $current\_autore->descrizione;

inserisco il ruolo ottenendo un vettore fatto così:

**$info\_autori[$current\_autore->autore\_id] = [**

**nome => ‘nome e cognome dell’autore corrente’**

**ruoli => [$current\_autore->autore\_id => descrizione del ruolo dell’autore corrente]**

**]**

Se non è la prima volta che inserisco quell’autore aggiungo solo eventuali ulteriori ruoli a quell’autore.

Ripeto il tutto finché ci sono autori da processare provenienti dalla query.

JQUERY

La libreria JQUERY consente di associare dei trigger a determinati eventi associati al DOM (Document object model) la struttura della pagina html.

La sintassi è la seguente:

*$(SELETTORE ELEMENTI).on(‘evento’, functon(){*

*CODICE DELLA FUNZIONE*

*});*

$ è l’identificatore della libreria JQUERY.

Un esempio è il seguente:

*$(‘.form.group’).on(‘change’, function () {*

*});*

Ogni volta che c’è un cambiamento in qualsiasi elemento della classe .form-group viene richiamata la funzione function ().

Ci sono vari eventi in genere per i form si usa l’evento change, mentre per gli elementi statici si usa l’evento click.

Vediamo un altro esempio:

<script>

$(document).ready(function(){

$('[name=data\_lettura]').datepicker({

format: 'dd-mm-yyyy',

todayHighlight: true

});

});

</script>

La prima riga $(document).ready(function () { ci dice che la funzione function verrà chiamata dopo che la pagina è stata caricata. La funzione associa all’elemento [‘name=data:lettura’] l’oggetto datepicker che è un calendario della libreria bootstrap.

* Format:’dd-mm-yyyy’ indica il formato della data visualizzata dal calendario
* todayHighlight: true evidenzia sul calendario la data corrente.